Job Cacher Plugin

Plugin Information
View Job Cacher on the plugin site for more information.

Summary

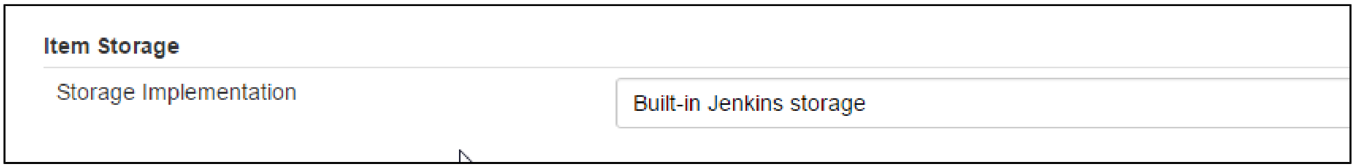
Enables caching of files on transient executors for use on subsequent builds to improve performance

Features

- Item storage extension point supporting on master storage and AWS S3
- Cache Wrapper for free style jobs that manages the cache
- Arbitrary File Cache where user specifies paths to be cached
- UI on Job page to review the job's caches
- Supports Pipeline jobs with a cache directive
- Cache Extension Point for other plugins to provide opinionated caching capability such as Gradle caches
- Incremental storing of cache into storage to minimize data movement

Global Configuration

By default, the plugin is configured to use on-master storage for the cache.

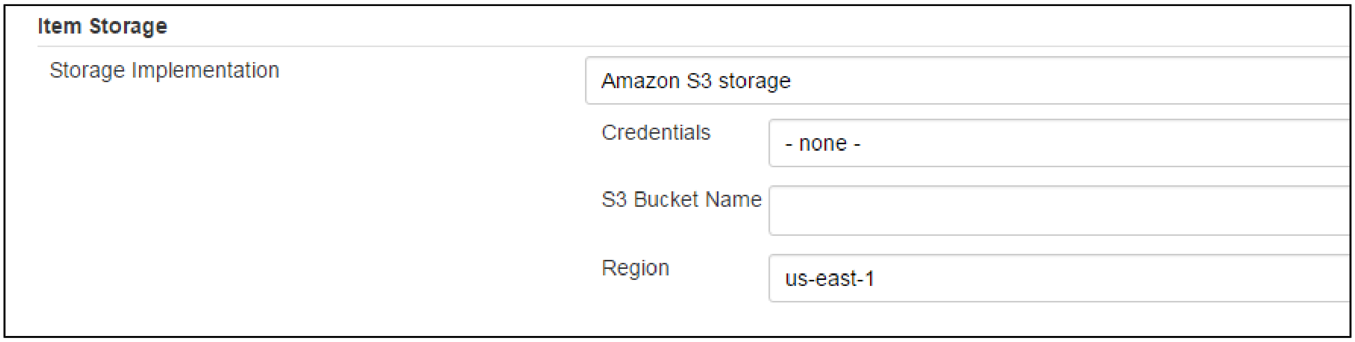


Item Storage

Storage Implementation: Built-in Jenkins storage

This storage mechanism is not recommended for heavy use as it will burden the remoting channel with considerable data when the job starts.

In addition to the on-master storage, a storage implementation for Amazon S3 is also available.



Item Storage

Storage Implementation: Amazon S3 storage

Credentials: - none -

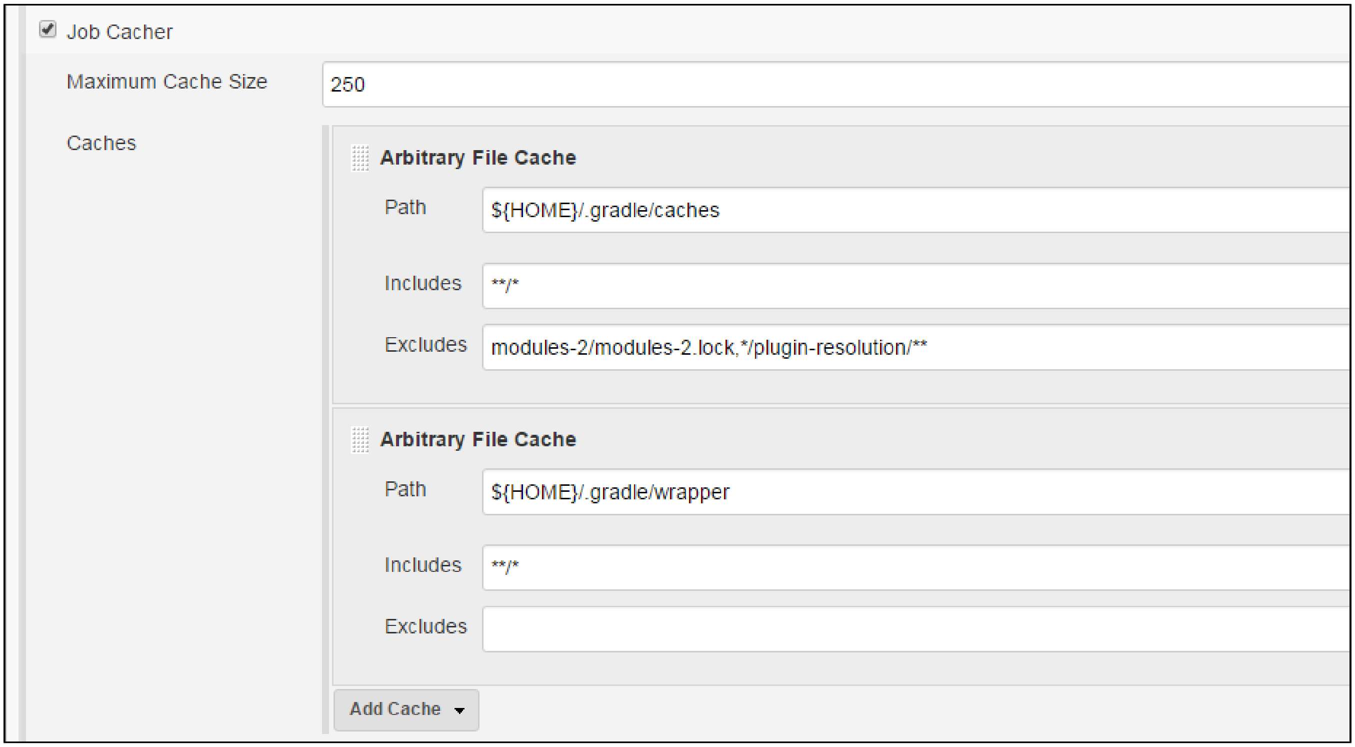
S3 Bucket Name:

Region: us-east-1

Additional storage implementations can be contributed as well via the *ItemStorage* extension point.

Job Configuration

The plugin offers a Build Wrapper extension point that can be configured for use in Free Style jobs. Prior to the build starting, the plugin checks if there is a populated cache and if so copies it from the storage area to the executor. At the end of the build, the plugin then copies the same cache, incrementally, back to storage.



☑ Job Cacher

Maximum Cache Size: 250

Caches

Arbitrary File Cache

Path: ${HOME}/.gradle/caches

Includes: **/*

Excludes: modules-2/modules-2.lock,*/plugin-resolution/**

Arbitrary File Cache

Path: ${HOME}/.gradle/wrapper

Includes: **/*

Excludes:

The above configuration caches the gradle cache and wrapper so that subsequent builds don't need to rebuild these folders and files through the dependency management repository. It also configures the cache to have a maximum size of 250 MB so that the cache won't grow indefinitely. Once this size is hit, the cache is deleted in the storage and the next build will need to repopulate the cache from scratch.

Pipeline Configuration

The plugin also supports the Pipeline plugin by introducing a cache build step that can be used within the pipeline definition.

```
 cache(maxCacheSize: 250, caches: [
     [$class: 'ArbitraryFileCache', excludes: 'modules-2/modules-2.lock,*/plugin-resolution/**', includes: '**
/*', path: '${HOME}/.gradle/caches'],
     [$class: 'ArbitraryFileCache', excludes: '', includes: '**/*', path: '${HOME}/.gradle/wrapper']
  ]) {
    // some block
}
```

Cache Types

The plugin currently ships with a single cache implementation called *ArbitraryFileCache* which must be explicitly configured with a path and includes / excludes rules to apply to find the files to cache. Since *Cache* is an extension point in the plugin, other plugins can contribute implementations that deliver specific behavior such as a *GradleCache* that could automatically look to cache all gradle related files that can be cached from build to build.

Visualizing the Cache

A job action is added to the job so that users can look at the contents of the cache.

Currently for on master storage, this is visualized through the Jenkins interface and for Amazon S3 storage, this redirects the user to the S3 console to view the cache contents.

Running it

As the job begins, an entry will be published to the console output for each cache to indicate the executor is being primed with the files from storage:

```
Caching /root/.gradle/caches to executor
Caching /root/.gradle/wrapper to executor
```

At the end of the job run, each cache is then copied back to the storage area

```
Storing /root/.gradle/caches in cache
Storing /root/.gradle/wrapper in cache
```

Other plugins supporting Job Cacher

Content by label

There is no content with the specified labels

Version History

1.0

- Initial version